Synthesizing Task Periods for Dwells in Multi-Function Phased Array Radars

Chi-Sheng Shih
Dept. of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan 106 ROC
cshih@csie.ntu.edu.tw

Phanindra Ganti
QualComm Inc.
San Diego, CA 92121 USA
pganti@qualcomm.com

Sathish Gopalakrishnan Marco Caccamo Lui Sha
Dept. of Computer Science
University of Illinois
Urbana, IL 61801 USA
{sgopalak, mcaccamo, lrs}@uiuc.edu

Abstract

This paper addresses the problem of scheduling radar dwells in multi-function phased array radar systems. The timing constraint of radar tasks are usually modeled by the minimal and maximal temporal distance between any two consecutive dwells of a task. Such a timing constraint makes it difficult for traditional real-time scheduling techniques to provide predicatable timing guarantee, without over-consuming the resources. We propose a novel approach to model the dwells as periodic real-time tasks. The periods of the tasks are synthesized by the minimal and maximal temporal distance constraint of the dwells. The synthetic periods allow the template-based scheduling algorithm to compute efficient dwell schedules with low overhead. We evaluate the algorithms via extensive simulations. Simulation results show that this algorithm can significantly improve the resource utilization, comparing with traditional dwell scheduling algorithms.

1 Introduction

In a multi-function phased array radar system, radar tasks search or track targets in its surveillance space. To keep track of the targets, the system must meet its timing and energy constraints. The timing constraint enables the system to illuminate (electro-magnetic) waves on the targets with high probabilities; the energy constraint prevents the system from overheating damage. The system fails when it fails to meet any of these two constraints.

A (radar) track task is an end-to-end task [1, 2] with three subtasks: a control command subtask, a dwell subtask, and a signal processing subtask [3, 4]. When a track task starts, the control command subtask generates the commands of sending/receiving the waves. Next, based on the generated commands, the dwell subtask sends and receives the waves at a scheduled time. Then, the signal processing subtask processes the collected signal to discover the location of the tracked target. The system uses the result of the signal processing subtask to predict the movement of the target in the future. The estimated location of the target is given to the next instance of the track task and the execution sequence of the three subtasks repeats. The track task stops when there is no need to monitor the movement of the target.

In this paper, we focus on scheduling dwell subtasks on the radar antenna. This is because the executions of radar dwells are often the performance bottleneck of radar systems. In addition, the cost of radar antenna are often too high to add as many as possible to improve the performance. However, the performance of the other two subtasks can be optimized by adding more processors to the systems.

While sending the dwells, the system must maintain the minimal and maximal temporal distance between any two consecutive dwells of a track task. Figure 1 gives an illustrative example.

In the figure, the upper portion shows the trajectory of the tracked target and the n and $(n+1)$-th dwells and the lower portion shows the time line. Suppose that the n-th dwell completes at time a. *Processing window* $(a,b]$ is the time interval for executing the signal and control command subtasks. To assure that the two subtasks execute completely, the $(n+1)$-th dwell cannot be scheduled in interval $(a,b]$. The difference $b-a$ is the *minimal temporal distance* of the dwells. The coverage of the waves and the estimates of the target location are shown as the shaded and dashed circles in the figure, respectively. Because the uncertainty of the velocity of the target increases as time goes on, the estimates of the target location also grow as time goes on. Within time interval $(b,d]$, the coverage of the waves always contain the estimates of the target location. Therefore, when the $(n+1)$-th dwell is sent after time d, the waves may miss the target with a high probability. The difference $d-a$ is called the *maximal temporal distance* of the two dwells and the interval $(b,d]$ is the *illumination window* of the $(n+1)$-th dwell.

Traditionally, the system either disregards the actual timing constraints (e.g., the best effort scheduling algorithms [4]) or uses the most stringent values just to be safe (e.g., using the half length of the maximal temporal distance as the period of periodic tasks.) As exemplified by Figure 1, the dwells of a track task may not be released at a constant time interval. The result is unpredictable timing behavior and/or poor resource utilization. In addition, the "estimated" scheduled time of the $(n+1)$-th dwell must be known before the control command subtask starts. Otherwise, the generated commands could be problematic. Traditional dynamic real-time scheduling algorithms such EDF algorithm makes it difficult to predict the scheduled time be-

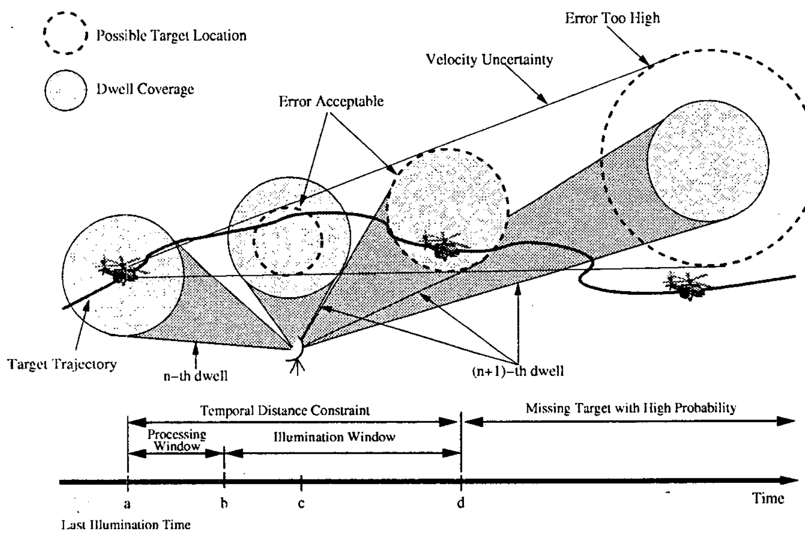


Figure 1. Temporal Distance Constraint

fore a job is released.

Our work is related to the works dealing with temporal distance constraints (e.g., [5, 6, 7, 8]). In these works, early completions are acceptable. However, as exemplified by Figure 1, early completion fail the tasks. Kuo et al. in [9] propose a reservation-based approach for real-time dwell scheduling. This approach allows the system to guarantee the performance requirement when the schedulability condition holds. However, Kuo et al. assume that the sampling periods of tasks are known and do not consider the energy constraint.

We extend the template-based scheduling algorithm proposed by [3, 10] and modify the idea of "synthetic period." Note that, according to [3, 10], the periods of dwell tasks are assigned so that the tasks can be grouped based on their assigned periods. Hence, the template-based scheduling algorithms are able to off-line construct the dwell schedule for the performance requirement task set. However, this approach only allows relatively small jitters on the temporal distance. As a result, when the system workload is high, the template-based scheduling algorithm may not be able to find a feasible schedule. To overcome this problem, the key idea of this paper is to determine a time interval for every dwell so that the dwell can be scheduled at any time within the time interval without considering the completion time of its preceding dwell. The template-based scheduling algorithm constructs the scheduling without grouping the tasks based on their periods.

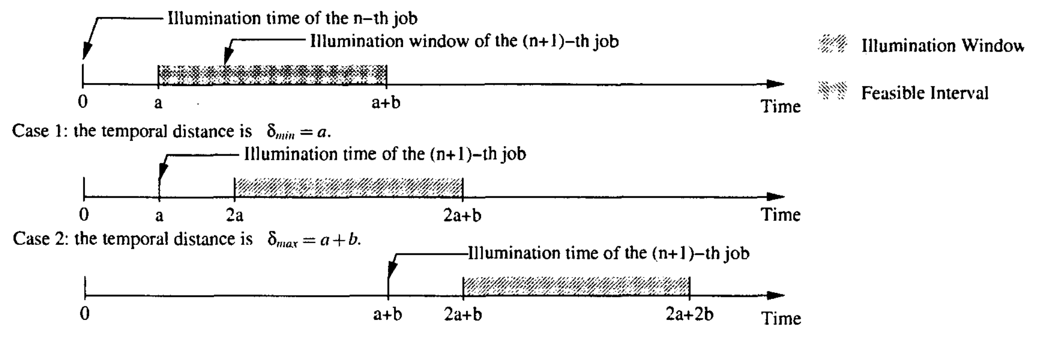
Section 2 that follows describes the task model and defines the terms used here. The section also states the problem of scheduling radar dwells. Section 3 presents the period assignment algorithm to determine the synthetic period of each task. We also present the algorithm of transforming dwell tasks into periodic tasks with synthetic periods. Section 4 presents the algorithms conducting the schedulability analysis and constructing the template schedule. Section 5 evaluates the algorithms against other scheduling algorithms such as template-based scheduling algorithm [3, 10] via simulations. Lastly, Section 6 summarizes the paper.

2 Formal Model and Problem Statement

Formal Model A *dwell pattern*, called a *dwell* for short and denoted by W, sends and receives electro-magnetic waves. A dwell is defined by its execution time and power function. The execution of a dwell consists of three sequential phases: *sending*, *round-trip delay*, and *receiving phase*. During the sending and receiving phases of a dwell, the system should not preempt the dwell. A dwell fails when it is preempted in these two phases. The radar system could be idle or execute other dwells during the round-trip delay phase of one dwell. The execution time of a dwell W, denoted by e_W, is the sum of the times to execute these three phases of dwell W. The amount of power consumed by a dwell is described by a time function. A power function $P_W(t)$ of dwell W for $0 \leq t \leq e_W$ denotes the consumed power at the elapsed time t since the dwell has started to execute. During the round-trip delay phase, a dwell consumes no power and the values of the power function are zero. A dwell is defined as $W = (e_W, P_W(t))$.

Thus far, and in our subsequent discussion, we use the terms task and job as they are commonly used in real-time systems literature [11, 5, 12]. A *job* is an instance of the illumination of a radar dwell. A *task* is a sequence of jobs that have identical or similar characteristics and timing requirements. We call tasks T_1, T_2, etc. In addition, we call the j-th instance of task T_i job $J_{i,j}$. Except for where it is stated otherwise, by a task and a job, we mean a dwell subtask and a job of a dwell subtask, respectively.

The timing parameters of a task T_i includes its release time, execution time, minimal temporal distance and maximal temporal distance. The *release time* of task T_i, denoted



Illumination time of the n–th job
Illumination window of the (n+1)–th job
Illumination Window
Feasible Interval
0
a
a+b
Time
Case 1: the temporal distance is $\delta_{min} = a$.
Illumination time of the (n+1)–th job
0
a
2a
2a+b
Time
Case 2: the temporal distance is $\delta_{max} = a + b$.
Illumination time of the (n+1)–th job
0
a+b
2a+b
2a+2b
Time

Figure 2. Illumination Windows

by r_i, is the instant of time at which the task becomes known to the system. The execution time, denoted by e_i, is the time required to send and receive the waves. The minimal temporal distance and maximal temporal distance of task T_i, denoted by $\delta_{i,min}$ and $\delta_{i,max}$, is the lower and upper bound for the temporal distance of its jobs, respectively. The *illumination time* of a job is the instant of time at which the radar system starts sending the radar beams for the job. The *illumination window* of job $J_{i,j}$ is a time interval which starts $\delta_{i,min}$ units of time after the illumination time of its preceding job $J_{i,j-1}$ and whose length is the difference $\delta_{i,max} - \delta_{i,min}$. For the first job of a task, its illumination windows starts at $\delta_{i,min}$ units of time after the release time of the task.

A *feasible interval* of a job is the time interval during which the job always meets its temporal distance constraint if it executes from start to completion in the interval. When the start time of the preceding job of a job is known, the feasible interval can be equal to the illumination window of the job.

Timing Constraint The following definition states the timing constraint of a dwell job.

Definition 2.1 (In-Time Completion of Dwell Jobs). *A dwell job executes to completion if there is no interruption during its sending and receiving phases. A dwell job completes in time if and only if it executes to completion within its illumination window.*

A dwell task completes in time if all its jobs complete in time. The following definition states the timing constraint of a dwell task.

Definition 2.2 (In-Time Completion of Dwell Tasks). *A dwell task completes in time if and only if all of its dwell jobs complete in time.*

Problem Statements The performance requirement of a radar system is often given as a set of search and track scenarios. A scenario describes how many search and track tasks it should be able to execute simultaneously in order to monitor the surveillance space. Such a scenario can be transformed into a radar task set, denoted by $\mathbf{T} = \{T_1^*, T_2^*, ..., T_N^*\}$, in which radar tasks are indexed in ascending order of their maximal temporal distances. In this paper, we assume there is only one performance requirement for the system and only one radar antenna in the system. The real-time dwell scheduling problem is stated as follows: We are given the performance requirement $\mathbf{T} = \{T_1^*, T_2^*, ..., T_N^*\}$ and the energy threshold E_{TH} for a look-back period τ. In addition, non-critical dwell task T_j for $j > N$ may arrive at any time after the system starts. The problem is to find a scheduling algorithm so that all the tasks in set $\mathbf{T}$ meet their timing constraints and there is no energy constraint violation for all times. In addition, we want to schedule as many non-critical tasks as possible.

3 Period Synthesis and Dwell Scheduling

3.1 Preliminary Discussion

According to the definition of feasible intervals, a job meets its timing constraint if it is scheduled to start within its feasible interval. However, assigning the feasible intervals for jobs is not trivial. Figure 2 gives an illustrative example.

Suppose the n-th job starts to illuminate at time 0 and the minimal and maximal temporal distance are a and $a+b$, respectively. The illumination window of the $(n+1)$-th job is interval $[a, a+b]$ and can be set as its feasible interval, shown as the cross-hatched box in the first time line in the figure. However, computing the feasible interval of the $(n+2)$-th job is not trivial. Although $(n+1)$-th job can be scheduled at any time in interval $[a, a+b]$, it is sufficient to consider only two cases: the job start at time a and time $a+b$. This is because the the length of the illumination window is a constant, which is b in this example. When the $(n+1)$-th job starts to execute at time a, the illumination window of the $(n+2)$-th job is interval $[2a, 2a+b]$, shown on the second time line in the figure. When the $(n+1)$-th job starts to execute at time $a+b$, the illumination window of the $(n+2)$-th job is interval $[2a+b, 2a+2b]$, shown on the third time line in the figure. The feasible interval of the job must be the overlapping region of all the possible illumination windows. Unfortunately, there is no overlap between these two illumination windows, as shown in the figure. In other words, if the $(n+1)$-th job is allowed to start at any time in its illumination window $[a, a+b]$, the length of the feasible interval of the $(n+2)$-th job is zero.

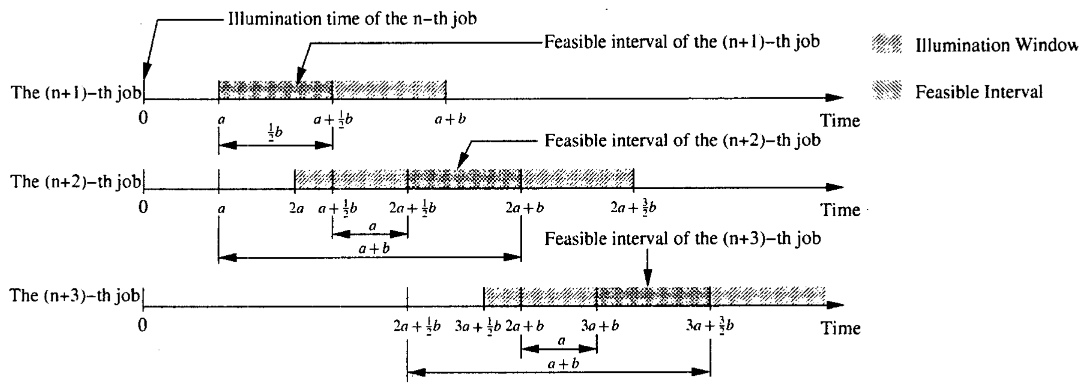


Figure 3. Synthetic Periods

3.2 Period Synthesis Algorithm

The key idea of the Period Synthesis Algorithm is to limit the jobs to be executed in certain time intervals so that the feasible intervals of the following jobs can be analyzed, without knowing the exact illumination times of the jobs. Figure 3 gives an illustrative example.

Again, the n-th job starts to illuminate at time 0 and the minimal and maximal temporal distance are a and $a+b$, respectively. Hence, the illumination window of the $(n+1)$-th dwell is $[a, a+b]$. Rather allowing the $(n+1)$-th job to start at any time in its illumination window, the Period Synthesis Algorithm limits the $(n+1)$-th job to start in interval $[a, a+\frac{b}{2}]$, shown as the cross-hatched box on the first time line in the figure. Because interval $[a, a+\frac{b}{2}]$ completely overlaps with the illumination window of the $(n+1)$-th job, the interval can be set as the feasible interval of the $(n+1)$-th job. When the $(n+1)$-th job starts at time a, the illumination windows of the $(n+2)$-th job is $[2a, 2a+b]$; When the $(n+1)$-th job starts at time $a+\frac{b}{2}$, the illumination windows of the $(n+2)$-th job is $[2a+\frac{1}{2}b, 2a+\frac{3}{2}b]$. As shown on the second time line in the figure, these two illumination windows overlap at interval $[2a+\frac{1}{2}b, 2a+b]$, which can be set as the feasible interval of the $(n+2)$-th job. The figure also shows that the distance of any two time instants in these two feasible intervals are bounded by a and $a+b$. Hence, when the $(n+1)$-th and $(n+2)$-th jobs start to execute at any time within their corresponding feasible intervals, the temporal distance is bounded by a and $a+b$. Similarly, the feasible interval of the $(n+3)$-th job is interval $[3a+b, 3a+\frac{3}{2}b]$, shown on the third time line in the figure. The feasible intervals have constant length $\frac{b}{2}$ and repeat for $a+\frac{b}{2}$ units of time, which is called the *synthetic period* of the task. Hence, the task can be modeled as a periodic task whose jobs are released at a constant interval $a+\frac{b}{2}$.

Period Synthesis Algorithm. *Given a task set* $\mathbf{T} = \{T_1^*, T_2^*, ..., T_N^*\}$, *the Period Synthesis algorithm computes the period and relative deadline for each task* T_i^* *in the following steps:*

Step 1. *For each task* $T_i^* = \langle W_i, \delta_{i,min}, \delta_{i,max}\rangle$, *the Period Synthesis algorithm transforms task* T_i^* *to a periodic task whose relative deadline is* $d_i = \frac{\delta_{i,max}-\delta_{i,min}}{2}$ *and synthetic period is* $s_i = \frac{\delta_{i,max}+\delta_{i,min}}{2}$.

Step 2. *For each synthetic period* s_i *for* $1 \le i \le N$ *[5],*

2.1. *The harmonic base* b_i *is defined as* $b_i = s_i/2^{\lceil \log_2 \frac{s_i}{s_1} \rceil}$.

2.2. *With respect to the harmonic base* b_i, *the specialized period of task* T_j *for* $1 \le j \le N$ *is* $P_{j,i} = b_i \times 2^{\lfloor \log_2 \frac{s_j}{s_1} \rfloor}$.

2.3. *The total instantaneous utilization for harmonic base* b_i *is* $u_i = \sum_{j=1}^{N} e_j/P_{j,i}$.

Step 3. *Select the period* $P_j = P_{j,i}$ *for each task* T_j *for* $1 \le j \le N$ *such that the total instantaneous utilization for base* b_i, *i.e.,* u_i, *is the minimal.*

We show that the synthetic period s_i computed at Step 1 is the optimal.

Theorem 3.1. *Given a dwell task* $T_i^* = \langle W_i, \delta_{i,min}, \delta_{i,max}\rangle$, $d_i = \frac{\delta_{i,max}-\delta_{i,min}}{2}$ *is the maximal constant relative deadline and* $s_i = \frac{\delta_{i,max}+\delta_{i,min}}{2}$ *is the maximal constant period for the task.*

Proof: The theorem follows straightforwardly. ■

3.3 Dwell Scheduling

Schedulability is tested by exact analysis, i.e., constructing the schedule of the tasks for the hyper-period. The template-based scheduling algorithm [3, 10] constructs a library of templates which optimize the schedule of a set of jobs in a short time interval to reduce the residual heat, and uses the templates as the building blocks to efficiently construct the schedule. The Synthetic Period Template-Based (SPTB) algorithm is an extended template-based scheduling algorithm. The scheduling priorities of jobs are defined by the absolute deadlines of the jobs. The earlier the deadline, the higher the priority. When selecting the template,

Table 1. Dwell Task Parameters

Tasks	Importance	Execution time (ms)	Average power consumption (kW)	(C_{min}, C_{max})(ms)
High Priority Search	1	(1, 4, 1)	(5, 0, 0.1)	(500, 1000)
Confirmation Task	2	(1, 4, 1)	(4, 0, 0.1)	(200, 800)
High-Precision Track	3	(0.5, 1, 0.5)	(4, 0, 0.1)	(50, 180)
Precision Track	4	(1, 2, 1)	(4, 0, 0.1)	(250, 600)
Normal Track	5	(1, 2, 1)	(3, 0, 0.1)	(850, 1000)
Low-Priority Search	6	(0.5, 1, 0.5)	(3, 0, 0.1)	(750, 1800)

the SPTB algorithm uses the first-fit bin-packing algorithm [13]. In the SPTB algorithm, the jobs scheduled in one template may not have the same period, which allows the system to utilize the resource better. When there exists a schedule for the given task set, the system can repeat the schedule for every hyper-period of the task set. Both the energy and timing constraints are met.

4 Performance Evaluation

We evaluated the Period Synthesis and SPTB scheduling algorithm by extensive simulations and compared their performance against that of the priority-driven best-effort algorithm [4] and Least Template Slack First (LTSF) template-based scheduling algorithm [10].

The parameters for radar dwells are based on the parameters described by Kuo et al. [9] and similar to the parameters used in [3] and [10]. Search tasks always present in the system. Each search task can potentially generate one confirmation task. On completion of a confirmation task, track tasks are generated based on a uniform distribution for the results of the confirmation task. A confirmation task can generate a track task with probability p_t, varied from 0.1 to 0.8. The track task can be a high-precision track with probability p_{hpt}, a precision track with probability p_{pt}, or a normal track with probability p_{nt}, such that $p_{hpt} + p_{pt} + p_{nt} = 1$. In our simulations, we set $p_{hpt} = p_{pt} = p_{nt}$.

The workload parameters are listed in Table 1. The three element tuples for execution time and power consumption denote the time and average power consumption for the sending, round-trip delay and receiving phases of a dwell task. The pair (C_{min}, C_{max}) denotes the minimal and maximum temporal distance between the dwells of a task.

The performance requirement of the simulated system is 45 high priority search tasks, 10 confirmation tasks, 10 high-precision tracks, 10 precision tracks, 5 normal tracks and 20 low-priority search tasks. The system should be able to execute all these tasks simultaneously. Tasks not in the performance requirement are scheduled based on their semantic importance. The energy threshold is $250J$ and with an energy look-back period of $200ms$.

We use three performance metrics to measure the performance of our algorithms: mean utilization, mean rejection rate, and mean (admission control) overhead. The *mean utilization* is the fraction of time the antenna is either sending or receiving electro-magnetic beams. The *mean rejection rate* refers to the fraction of dwells that do not complete in time. They include tasks that fail the admission control test or are dropped because a critical task arrives. The *mean overhead* is the amount of time being used to conduct the schedulability analysis.

4.1 Results and Discussion

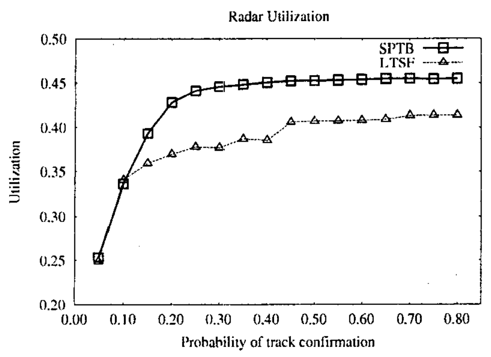
The result shows that the proposed approach increases the mean utilization and reduces the rejection rates. Note that the simulated system may not be able to achieve 100% utilization. This is because the system has to meet the energy constraint, which may limit the maximal utilization of the antenna. Comparing with the LTSF template-based scheduling algorithm, the mean utilizations have been increased by 5% to 10% as shown in Figure 4(a). Note that the LTSF template-based scheduling algorithm by itself increases the mean utilization by 15% to 20%, comparing with the best-effort algorithm [10]. In other words, without replacing the physical antenna, the SPTB scheduling algorithm can track more targets simultaneously and the system meets the energy and timing constraints. The mean rejection rate have been reduced up to 20% as shown in Figure 4(b) and (c). The performance of tracking other targets show the similar results and are not plotted.

The trade-off is the overhead of admission control. The PSTB algorithm requires more time than the LTSF template-based scheduling algorithm to conduct the schedulability analysis. However, as shown in Figure 4(d), the mean overhead is less than $1ms$ on our simulation machine which uses one Intel Pentium 4 2GHz processor and 512 MB memory. In practice, more processors can be added to reduce the admission control overhead.

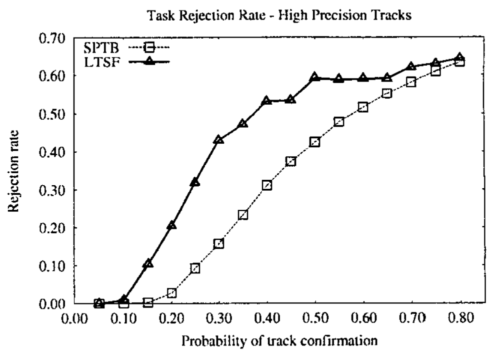
5 Summary

We presented here the dwell job model that characterizes real-time radar dwells. A radar dwell has to complete within some time interval to meet its timing constraint. We developed an algorithm to synthesize tasks with non-harmonic periods using tasks with harmonic periods. The synthetic periods allow the system to decouple actual task periods (harmonic or not) from the temporal distance constraints. We also developed the Fewer-Template-Slack-First template scheduling algorithm to schedule dwell jobs in their feasible intervals.

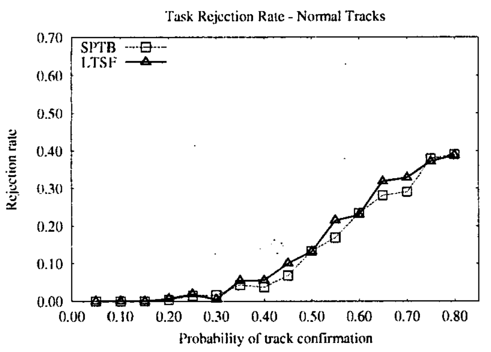
We evaluated the proposed template scheduling algorithm by extensive simulations and compared their performance against that of the LTSF and best efforts algorithm. The result shows that SPTB algorithm increases the resources utilization and reduces the rejection rates. In other words, using tasks with synthetic periods better utilizes the



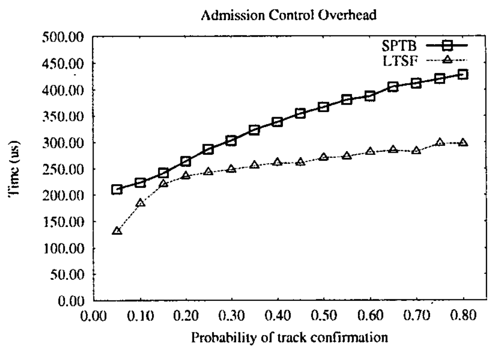
(a)



(b)



(c)



(d)

Figure 4. Performance Evaluation Results

system resources and the algorithm can effectively schedule tasks to meet their timing constraints.

Acknowledgment This work is supported in part by a grant from the MURI program N00014-01-0576, in part by ONR N0004-02-0102, in part by Lockheed Martin Corporation 1-5-36137, and in part by NSF CCR-0237884.

References

[1] R. Bettati. *End-To-End Scheduling To Meet Deadlines In Distributed Systems.* PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1994.

[2] J. Sun. *Fixed-Priority End-to-End Scheduling in Distributed Real-time Systems.* PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1997.

[3] C.-S. Shih, S. Gopalakrishnan, P. Ganti, M. Caccamo, and L. Sha. Template-based real-time dwell scheduling with energy constraint. In *Proceedings of the IEEE Real-Time and Embedded Technology and Application Symposium*, 2003.

[4] R. A. Baugh. *Computer Control of Modern Radars.* New York: RCA Corporation, 1973.

[5] C. Han and K. Lin. Scheduling distance-constrained real-time tasks. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 300 – 308, Dec. 1992.

[6] C.-W. Hsueh, K.-J. Lin, and N. Fa. Distributed pinwheel scheduling with end-to-end timing constraints. In *Proceedings of the IEEE Real-Time Systems Symposium*, 1995.

[7] C.-C. Han, K.-J. Lin, and C.-J. Hou. Distance-constrained scheduling and its applications to real-time systems. *IEEE Transaction on Computers*, 45(7):814 – 826, July 1996.

[8] L. Dong, R. G. Melhem, and D. Mossè. Time slot allocation for real-time messages with negotiable distance constrains. In *Proceedings of the IEEE Real-Time Technology and Application Symposium*, pages 131 – 136, 1998.

[9] T.-W. Kuo, Y.-S. Chao, C.-F. Kuo, C. Chang, and Y.-L. Su. Real-time dwell scheduling of component-oriented phased array radars. In *Proceedings of the IEEE 2002 Radar Conferences*, pages 92 – 97a, April 2002.

[10] C.-S. Shih, S. Gopalakrishnan, P. Ganti, M. Caccamo, and L. Sha. Scheduling real-time dwells using tasks with synthetic periods. In *Proceedings of the IEEE Real-Time Systems Symposium*, 2003.

[11] C. L. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[12] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic task scheduling for hard-real-time systems. *Real-time Systems Journal*, July 1989.

[13] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W. H. Freeman, 1979.